Refining and Verifying the Solution of a Linear System*

Hong Diep Nguyen
INRIA
LIP (CNRS - ENS de Lyon - INRIA - UCBL)
Université de Lyon
ENS de Lyon
46 allée d'Italie
69007 Lyon, France
hong.diep.nguyen@ens-lyon.org

Nathalie Revol
INRIA
LIP (CNRS - ENS de Lyon - INRIA - UCBL)
Université de Lyon
ENS de Lyon
46 allée d'Italie
69007 Lyon, France
nathalie.revol@ens-lyon.fr

ABSTRACT

The problem considered here is to refine an approximate, numerical, solution of a linear system and simultaneously give an enclosure of the error between this approximate solution and the exact one: this is the *verification* step. Desirable properties for an algorithm solving this problem are accuracy of the results, complexity and performance of the actual implementation. A new algorithm is given, which has been designed with these desirable properties in mind. It is based on iterative refinement for accuracy, with well-chosen computing precisions, and uses interval arithmetic for verification.

Categories and Subject Descriptors

G.1.3 [**Numerical Linear Algebra**]: Error analysis - Linear systems (direct and iterative methods); G.4 [**Mathematical Software**]: Verification

General Terms

Reliability, verification.

Keywords

Scientific computing, numerical linear algebra, verified computations, symbolic-numeric, interval arithmetic, floating-point arithmetic, precision, accuracy.

1. INTRODUCTION

The solution of a linear system using floating-point arithmetic entails roundoff errors. One approach to get exactly the solution consists in representing the coefficients as rational numbers and in solving the linear system exactly. Another approach, developed here, consists in mixing iterative refinement – for a better accuracy – and interval arithmetic – for guarantee – to get a tight enclosure of the exact solution. This latter approach can extend to linear systems with interval coefficients of small width.

The algorithm proposed here evolved from an initial version detailed in Section 2, which is very similar to the best available implementation, namely the `verifylss` routine of IntLab [9]. However, this initial version had 3 flaws: failure when the matrix of the system is too ill-conditioned, loss of accuracy and increase of the execution time when it is ill-conditioned. Working on two of these three weak points led us to the algorithm given in Section 3. Further remarks open the way to a better understanding of the respective strengths and weaknesses of exact computations and interval computations, in Section 4.

2. PROBLEM AND INITIAL ALGORITHM

We consider numerical computations, in other words floating-point arithmetic. Let A be an $n \times n$ matrix and b an n-dimensional vector, both with floating-point coefficients: $A \in \mathbb{F}^{n\times n}$ and $b \in \mathbb{F}^n$. Let us denote by x^* the exact solution of the linear system $Ax = b$, and let $\tilde{x}$ be an approximate solution, computed with a so-called numerical routine (LU with partial pivoting in our experiments, such as the `dgetrf` routine of LAPACK). The vector $\tilde{x}$ has floating-point coefficients.

Let us denote by $e = x^* - \tilde{x}$ the error between the exact and the approximate solutions. We look for an enclosure of e, that is, a vector with interval coefficients $\mathbf{e}$ such that $e \in \mathbf{e}$. Let us note here that intervals are denoted with boldface characters. More precisely, the goal is to develop an algorithm which increases the accuracy of $\tilde{x}$ and computes $\mathbf{e}$. We would like this algorithm to satisfy the following four criteria and we will explain our contribution in this direction:

- $\mathbf{e}$ contains the error $e = x^* - \tilde{x}$;
- the accuracy of the result should be close to the best accuracy offered by the floating-point arithmetic, i.e. the relative error should be close to 2^{-53} in IEEE-754 double precision floating-point arithmetic;
- the complexity of the algorithm should be comparable to the complexity of the numerical algorithm, which is $\frac{2}{3}n^3$: the algorithm must exhibit a complexity $\mathcal{O}(n^3)$ and the constant hidden in the $\mathcal{O}$ must be moderate;
- the actual performance of the implementation should be close to the performance of the numerical routine.

The starting point is to use iterative refinement, and to replace floating-point operations by interval operations when

*Supported by the ANR project EVA-Flo.

SNC 2011, June 7-9, 2011, San Jose, California.

it is appropriate to do so. Indeed, iterative refinement is a method that starts from an approximate solution and "contracts" the error, and contractant iterations are methods of choice in interval arithmetic. An algorithm which uses this idea to refine the solution and enclose the error is the `verifylss` function of IntLab [9], a MatLab toolbox that offers interval arithmetic. Our initial algorithm, called `certifylss`, differs from `verifylss` in the computation of $\mathbf{e}$, the enclosure of the error: `verifylss` employs Krawczyk iteration [4] and the Hansen-Bliek-Rohn-Ning-Kearfott formula [5] whereas `certifylss` uses Gauss-Seidel or Jacobi [4, pp. 131 ff.]. (Let us mention here that computing exactly the solution of this interval linear system is not considered: it is known to be NP-hard [8]). The approximate solution $\tilde{x}$ is then corrected and the correction term is the center of $\mathbf{e}$, the enclosure of the error.

A main difference with numerical iterative refinement is that the algorithm premultiplies, using interval arithmetic, the matrix A of the linear system by an approximate, numerical, inverse R: the resulting system has a matrix which should be close to the identity matrix, or rather which should be an H-matrix (see [4, p. 111] for the definition) and thus which behaves well when it comes to solve a linear system.

Algorithm `certifylss` % in MatLab-like syntax
Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$
$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA]$
while(not converged)
  $\mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ % $RA(x^* - \tilde{x}) \in \mathbf{r}$
  $\mathbf{e} = \mathbf{K} \setminus \mathbf{r}$ % $x^* - \tilde{x} \in \mathbf{e}$
  $\tilde{x} = \tilde{x} + \text{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \text{mid}(\mathbf{e})$
end
Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

Actually, the iteration needs an initial enclosure $\mathbf{e}_0$ of the error, which is then refined at each step. Determining $\mathbf{e}_0$ is based on a heuristic, that relies on $\mathbf{K}$ being an H-matrix, and that can fail. Experiments indicate that failure occurs when A is ill-conditioned: our guess is that R then is a poor approximate of A^{-1} and that $\mathbf{K}$ is not an H-matrix.

The stopping criterion is reached either when the iteration stagnates (no change for $\mathbf{e}$) or when the approximate solution $\tilde{x}$ is approximate to the last bit, in other words when the width of $\mathbf{x}$ corresponds to a relative error of 2^{-52} on $\tilde{x}$, in IEEE-754 double precision.

As already mentioned, three flaws have been identified: loss of accuracy and large execution time for ill-conditioned matrices, failure of the algorithm for extremely ill-conditioned matrices. A new algorithm has been developed to obviate the first two flaws.

3. NEW ALGORITHM TO IMPROVE AND VERIFY THE SOLUTION

3.1 Reducing execution time: relaxation technique

The major contribution to execution time is the operation $\mathbf{e} = \mathbf{K} \setminus \mathbf{r}$. This is performed iteratively, using Gauss-Seidel method which is intrinsically sequential. Furthermore, interval Gauss-Seidel iterations cannot be expressed using LAPACK (floating-point and thus fast) routines. However, they boil down to a floating-point matrix-vector product when Jacobi iterations are used and when the matrix $\mathbf{K}$ is centered around a diagonal matrix. Thus, to reduce execution time, the matrix $\mathbf{K}$ is inflated into a matrix $\widehat{\mathbf{K}}$ that contains $\mathbf{K}$ (otherwise it would no more be guaranteed that $e \in \mathbf{e}$) and that is centered around a diagonal. This is justified by the fact that $\mathbf{K}$ is expected to be close to the identity matrix. Moreover, Jacobi iterations are used instead of Gauss-Seidel ones. The corresponding algorithm is called `certifylss_relaxed`.

3.2 Increasing the accuracy of the result: well-chosen computing precisions

It is well-known [3, ch. 12] (and already done in `certifylss`) that computing the residual $\mathbf{r}$ is subject to cancellation and must be performed using twice the working precision, for accuracy purpose. Following an idea given in [1], we also compute the current approximate solution $\tilde{x}$ using twice the working precision, and we prove in [6] that this yields a fully accurate result, that is, an error of relative width to 1 ulp. Carefully chosen formulas for the computation of $\mathbf{e}$ yield this accuracy without implying extra, useless computations: terms corresponding to 3 times the working precision are not explicitly computed. Details are given in [6] and are not developed below.

3.3 New algorithm: certifylssx

Algorithm `certifylssx`
Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$
$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA]$
$\widehat{\mathbf{K}} = \text{inflated}(\mathbf{K})$ % $\widehat{\mathbf{K}}$ is centered on a diagonal matrix
while(not converged)
  $\mathbf{r} = [b - \mathbf{A}\,\tilde{x}]$ % $\mathbf{r}$ in twice the working precision
  $\mathbf{r} = R\mathbf{r}]$
  $\mathbf{e} = \widehat{\mathbf{K}} \setminus \mathbf{r}$
  $\tilde{x} = \tilde{x} + \text{mid}(\mathbf{e})$ % $\tilde{x}$ in twice the working precision
  $\mathbf{e} = \mathbf{e} - \text{mid}(\mathbf{e})$
end
Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

The performance of this new algorithm, in terms of accuracy (maximal componentwise relative width of the final $\mathbf{e}$) and execution time, is shown on Figure 1, each dot being the average on 10 matrices of dimensions 1000×1000 generated using the `gallery (randsvd, ...)` command of MatLab. These curves illustrate that we succeeded in obviating the two target problems: we obtained a solution accurate to the last bit, and we verified this accuracy, for the whole range of validity of our algorithm. We succeeded in doing so with acceptable execution time: the theoretical overhead factor is 6 and the practical factor is 15. Finally, let us also note that when the algorithm fails, it does so quickly, without wasting computational time.

4. CONCLUSION, FUTURE WORK, COMPLEXITY COMPARISONS

The algorithms presented in Sections 2 and 3 succeeded in providing an accurate solution $\tilde{x}$ of a linear system, with a guarantee on this accuracy being given by an enclosure $\mathbf{e}$ of the error. However, these algorithms fail when the condition

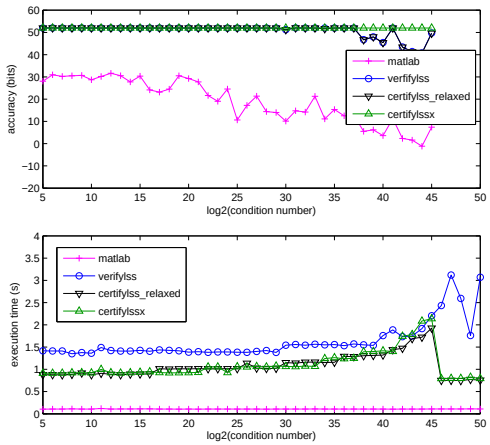

Figure 1: Performances of the `certifylssx` algorithm.

number of the matrix becomes too large. This is due to the fact that the approximate inverse R of A becomes poor, and thus the interval matrix $\mathbf{K}$ becomes wide. A solution would thus be to compute a more accurate R, using extended precision, as proposed in [10, 7]. Preliminary experiments tend to show that the increase in execution time is drastic. To keep it reasonable, again a solution seems to trade off execution time against accuracy in well-chosen parts of the code.

Desirable properties for such an algorithm were mentioned in Section 1: accuracy, complexity, execution time. Accuracy and execution time have been shown on figures and results are thoroughly proven in [6]. Let us mention the complexity of algorithms for solving linear systems:

- the numerical, non-verified algorithm (based on LU factorization with partial pivoting) has a complexity of $\frac{4}{3}n^3 + \mathcal{O}(n^2)$ flops (*flops* stands for *fl*oating-point *o*perations *p*er *s*econd);
- the numerical, verified algorithm of Section 3 has a complexity of $8n^3 + 2n^2p/(p - \log_2 \kappa(A)) + 8n^2 + \mathcal{O}(n)$ flops, where p is the precision (53 for the IEEE-754 double precision) and $\kappa(A)$ is the condition number of A. For the algorithm to terminate, one needs $\log_2 \kappa(A) < p$, in our experiments we need $\log_2 \kappa(A) < 45$. Because the refinement iterations converge linearly, as well as the iterations of $\mathbf{e}$ (Krawczyk iterations are known to converge linearly and Gauss-Seidel iterations are even faster [4]), one can show that the complexity of the iterative refinement part is $2n^2p/(p - \log_2 \kappa(A))$. These complexity results are new.
- the complexity of solving exactly the linear system [2], where the coefficients of A and b are considered as rational numbers, is $\tilde{\mathcal{O}}(pn^3)$ where again p is the precision, and $\tilde{\mathcal{O}}$ means that (poly-)logarithmic terms are not explicited.

In other words, this means that exact methods seem superior to get exact (and thus verified) results. However, only methods based on interval arithmetic can extend to data given with uncertainties, i.e. with coefficients which are (tight) intervals. Indeed, the algorithms given here apply, as long as $\mathbf{K}$ remains an H-matrix.

5. REFERENCES

[1] J. Demmel, Y. Hida, W. Kahan, X. S. Li, S. Mukherjee, and E. J. Riedy. Error bounds from extra-precise iterative refinements. *ACM Trans. on Mathematical Software*, 32(2):325–351, 2006.

[2] J. D. Dixon. Exact solution of linear equations using p-adic expansions. *Numerische Mathematik*, 40:137–141, 1982.

[3] N. J. Higham. *Accuracy and Stability of Numerical Algorithms (2nd ed.)*. SIAM, 2002.

[4] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, 1990.

[5] A. Neumaier. A simple derivation of the Hansen-Bliek-Rohn-Ning-Kearfott enclosure for linear interval equations. *Reliable Computing*, 5:131–136, 1999 (+ Erratum: Reliable Computing 6:227, 2000).

[6] H. D. Nguyen. *Efficient algorithms for verified scientific computing: numerical linear algebra using interval arithmetic*. PhD Thesis, École Normale Supérieure de Lyon, Jan. 2011. `http://perso.ens-lyon.fr/hong.diep.nguyen/`

[7] K. Ozaki, T. Ogita, and S. Oishi. An algorithm for automatically selecting a suitable verification method for linear systems. *Numerical Algorithms*, 56(3):363–382, 2011.

[8] J. Rohn. Enclosing solutions of linear interval equations is NP-hard. *Computing*, 53:365–368, 1994.

[9] S. M. Rump. INTLAB - INTerval LAboratory. In *Developments in Reliable Computing (Tibor Csendes ed.)*, pages 77–104. Kluwer Academic Publishers, 1999.

[10] S. M. Rump. Inversion of extremely ill-conditioned matrices in floating-point. *Japan Journal of Industrial and Applied Mathematics*, 26:249–277, 2009.